International Journal of Sciences: Basic and Applied Research (IJSBAR)

ISSN 2307-4531
(Print & Online)

http://gssrr.org/index.php?journal=JournalOfBasicAndApplied

---

# An Improved Neural Q-learning Approach for Dynamic Path Planning of Mobile Robots

Lei Ye[a]*, Siding Li[b], Zhenhua Huang[c]

[a,b,c]*College of Mechatronic Engineering and Automation, National University of Defense Technology, ChangSha, 410073, Hunan, China*

[a]*Email: lei.ye@foxmail.com*

[b]*Email: lsd0323@163.com*

[c]*Email: huangzhenhua_001@163.com*

**Abstract**

Dynamic path planning is an important task for mobile robots in complex and uncertain environments. This paper proposes an improved neural Q-learning (INQL) approach for dynamic path planning of mobile robots. In the proposed INQL approach, the reward function is designed based on a bio-inspired neural network so that the rate of convergence of INQL is improved compared with the Q-learning algorithm. In addition, by combining the INQL algorithm with cubic B-spline curves, the robot can move to the target position successfully via a feasible planned path in different dynamic environments. Simulation and experimental results illustrate the superior of the INQL-based path planning method compared with existing popular path planning methods.

***Keywords:*** Reinforcement learning; Q-learning; mobile robot; dynamic path planning.

## 1. Introduction

As an important class of machine learning methods, reinforcement Learning (RL) [17] has been popularly studied to solve sequential decision-making problems with uncertainties.

---

* Corresponding author.

A sequential decision-making problem is often modeled as a Markov decision process (MDP) [25] when applying a RL approach. The Q-learning algorithm [1] is a classic and widely used RL approach. Due to its simple application, Q-learning has been widely used in the areas of robot control [21, 22], multi-robot decision-making [23, 24], autonomous vehicle control [18] and elevator dispatching [19,20], etc.

In recent years, RL becomes a popular scheme applied in the area of robot path planning. As a simple table-type RL method characterized with simple structure and easy application, Q-learning is preferred to be used to solve path planning problems of mobile robots. Path planning problems of mobile robots [2] are often divided into two kinds: one is point-to-point path planning; the other one is the path planning of complete coverage. The point-to-point path planning is the most common, i.e., planning an optimal or near-optimal collision-free path from the starting point to the terminal point in the environment. The path planning of complete coverage is a special path planning in the two-dimensional space, i.e., a continuous optimal or quasi-optimal collision-free path covering all reachable areas is planned for the robot in the designated environment. Amit Konar [3,4] proposes an improved Q-learning method to solve the path planning problem of the single robot, which reduces the convergence time during the learning process and promotes the effectiveness of algorithm by introducing the bit zone. The robot path planning usually produces the continuous high-dimensional state spaces. As a table-type RL method, Q-learning usually suffers with the curse of the dimensionality in large-scale or high-dimensional state spaces. Some improved methods [5-7] which combine the neural network (NN) with the Q-learning algorithm are proposed to improve its approximation and generalization abilities in large-scale or high-dimensional state spaces. In addition, RL methods using value function approximation (VFA) such as LSPI and KLSPI [8,9] can be combined with the global path planning algorithms such as A* and PRM to solve the local path planning problem of the mobile robot.

When solving the MDP with large state and action space, the Q-learning algorithm is often confronted with the problems of large storage space, low learning efficiency, and slow convergence speed. During the learning process, user's feedback information or other additional guidance information could be provided to accelerate the learning process. The most common way is to record the action information of "expert" for executing the task and guide the research process of RL strategy with the information [10]. The reverse RL [11] method can be used to reversely deduce reward function via the track information. By deducing the reward function capable of punishing the action of deviating from the expected track, it can reproduce the true and unknown reward function of the "expert". Reward Shaping [12] is an approach which is often used to provide additional information to the learning agent based on the designed reward functions. The convergence rate of RL methods can be improved if the reward shaping approach is utilized.

In this paper, by combining the bio-inspired neural network with the Q-learning algorithm, an improved neural Q-learning (INQL) approach for dynamic path planning of mobile robots. In order to verify the effectiveness of INQL, other improved Q-learning algorithms are compared when learning in the environment with concave obstacles. The simulation results indicate that INQL algorithm has great advantages in the rate of convergence. In order to verify the effectiveness of the proposed dynamic planning method, simulations and experiments are conducted in different dynamic environments. The superior of the path planning method based on INQL is verified compared with existing popular path planning methods.

The rest of this paper is organized as follows. Section 2 provides some background introduction. Section 3 presents the Improved Neural Q-learning (INQL) algorithm for dynamic path planning. Section 4 gives simulation and experimental results to show the effectiveness of the proposed method. Section 5 draws conclusions and our future work.

## 2. Research Background

### *2.1. Markov Decision Process*

A Markov decision process (MDP) includes a state set $S$, an action set $A$, a state transition equation $T(s,a,s')$ and a reward function $R(s,a,s')$. The state set $S$ is a set of finite states $\{s_1, s_2, ..., s_N\}$, with the size equal to the total number of all states in the state set $N$. In terms of the state transition function, under the current state s, the system will execute the action $a$ to reach a new state $s'$, the state transition function is defined as $T: S \times A \times S \to [0,1]$. The reward function is the reward signal or punishment signal obtained from the environment when the intelligent agent executes action $a$ in State $s$ to reach the new State $s'$. The reward function is an important part of MDP model which influences the learning efficiency. Value functions can connect the optimal evaluation criterion and strategy, most of learning algorithms for solving MDP problems develop the optimal strategy by learning value functions and value functions indicate the status (good or bad) of the intelligent agent in the current state. The "status (good or bad)" is an evaluation criterion, which represents different meanings in different problems. The value function located at State s and subsequently selecting the action in accordance with the strategy $\pi$ is defined as $V^{\pi}(s)$; the value function located at State *s*, selecting Action *a* and subsequently selecting action in accordance with the policy $\pi$ is defined as $Q^{\pi}(s,a)$.

$$V^{\pi}(s) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s\right]$$

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a\right]$$

Where $0 < \gamma < 1$ is the discount factor, $E^{\pi}[\square]$ refers to the strategy $\pi$ and the expected value of state transition probability, and $\gamma_t$ is the reward value at the time *t*.

The state value function of the optimal strategy $\pi^*$ is defined as:

$$V^*(s) = E_{\pi^*}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| s_0 = s\right].$$

Where $r(s,a)$ is the reward value obtained after the execution of action $a$ in state $s$.

The value function of state-action pair of the policy $\pi$ is defined as:

$$Q^{\pi}(s,a) = E^{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t} r_{t+1} \mid s_0 = s, a_0 = a\right]$$

The optimal value function of state-action pair is:

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

After obtaining $Q^{*}(s,a)$, the optimal strategy $\pi^{*}$ can be obtained via:

$$\pi^{*}(s) = argmax Q^{*}(s,a)$$

### *2.2. The Bio-inspired Neural Network Model*

Inspired by the cell membrane model of the biological nervous system [13] of Hodgkin and Huxley and the shunting model [14] of Grossberg, Simon.Y [15] proposes a new neural network method to solve the path planning problem of the mobile robot. The neural network model is a topological structure, which indicates the state space of the mobile robot in the coordinate system developed by Descartes. The neural dynamics of each neuron is represented by a shunting equation or a simple addition equation and there is only local and horizontal excitatory junction in neurons. The complexity of calculation depends on the number of neurons in the network. In the aforesaid neural network, the target state attracts the mobile robot in the state space through the communication via the neural activity, and obstacles prevent the robot from collision with other objects within the limited spatial scope, Thus neurons can inhibit the horizontal connection of the neural activity.

When the bio-inspired neural network method is used, the dynamic characteristic of neuron *i* is represented by the shunting equation of neurons in Grossberg's shunting model. The received stimulating information comes from the target state and its horizontally connected adjacent neurons. Meanwhile, the stimulating information of neurons that does not directly connect with it cannot be received and the received inhibitory information only comes from the state of adjacent obstacles. As each neuron only receives the stimulating signals transmitted by its adjacent neurons, the inhibitory information of obstacles can be only delivered within the limited spatial scope.

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)([I_i]^{+} + \sum_{j=1}^{k}\omega_{ij}[x_j]^{+}) - (D + x_i)[I_i]^{-}$$

where *k* refers to the number of adjacent neurons of Neuron *i*.

The neural network model can ensure that the stimulating information sent from the target state can be delivered to all states in the work space through the horizontal connection of neurons. The inhibitory information sent from obstacles can be only transmitted within the limited spatial scope.

## 3. The Improved Neural Q-learning approach to dynamic path planning

### *3.1. MDP Modeling for Path Planning*

In the research on the path planning of the robot, there are many representation methods related to the environment and there are two models used for representing the indoor environment: geometric based model and topological model. Wherein, in the geometric model, the map is mainly represented in the form of grid and the representation method based on environmental features is applied on the basis of grid map. The grid map representation method is characterized in the little amount of computation and easy operation, thus it is frequently applied in the path planning of the mobile robot.

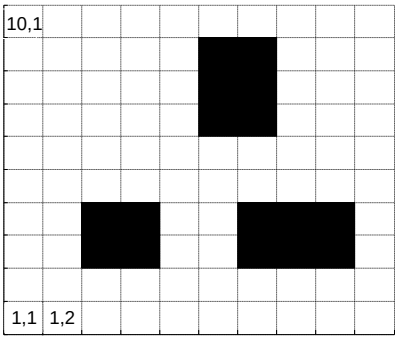



**Figure 1: Environmental Information of Grid Map**

In this paper, for the convenience of representing the information on the state and obstacles, the grid method is used to represent the environmental information. As shown in Figure1, each grid represents a state and its position on the map represents its state value. For example, for grids (1,1), (1,2) and (10,1), the black regions represent that the state information is an obstacle. As a constituent part of MDP, the action set significantly influences the learning efficiency and performance evaluation. Under general conditions, in order to promote the learning efficiency and algorithm performance, the action set will not be too large. In this paper, the optimal evaluation standard of the path planning problem is generally the length of planned path and the steering angle of the mobile robot; thus, the size of the action set is defined as 8, i.e., $A=\{a_1,a_2,...,a_8\}$, wherein, $a_1$=1, $a_2$=2,..., $a_8=8$, as shown in Figure 2.

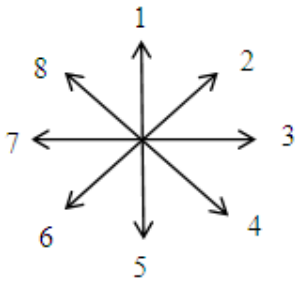



**Figure 2: Action Set**

The system executes Action $a$ in State $s$ and then transfers to the next State $s'$. Assumed State $s=(x,y)$, the state transition function is as follows $T(s,a)$:

$$T(s,a)=\begin{cases} s+(0,1), & a=1 \\ s+(1,1), & a=2 \\ s+(1,0), & a=3 \\ s+(1,-1), & a=4 \\ s+(0,-1), & a=5 \\ s+(-1,-1), & a=6 \\ s+(-1,0), & a=7 \\ s+(-1,1), & a=8 \end{cases}$$

The reward function is an important factor influencing the RL method efficiency and the result of reward function design directly influences the convergence rate of the algorithm. When the traditional Q-learning algorithm is used, the reward value at the target state is generally set as a positive integer, the reward value in the non-reasonable state(with any obstacle or out of bound) is defined as a negative integer and reward values in other states are all 0. As the reward function exerts no heuristic effect on the learning process of the robot, actions can be only selected randomly at the initial stage of learning. When both the state space and the action space are very large, it is slow for the robot to arrive at the target state. Thus, the design of the reward function must be instructive and the robot can arrive at the target state much faster.

In this paper, we propose a new reward design method which is based on the bio-inspired neural network method in [15]:

$$\frac{\Delta x_i}{\Delta t}=-Ax_i+(B-x_i)([I_i]^+ +\sum_{j=1}^{k}\omega_{ij}[x_j]^+)-(D+x_i)[I_i]^-$$

In case of $\Delta t=1$, $A=10$, $B=D=1$, $I=100$, the result of the aforesaid formula is as follows:

$$\Delta x_i=-10x_i+(1-x_i)([I_i]^+ +\sum_{j=1}^{k}\omega_{ij}[x_j]^+)-(1+x_i)[I_i]^-$$

The following result is recorded:

$$H=[I_i]^+ +\sum_{j=1}^{k}\omega_{ij}[x_j]^+$$

$$G=[I_i]^-$$

The following result is obtained:

$$\Delta x_i=-10x_i+(1-x_i)H-(1+x_i)G$$

The following iterative formula is obtained after the aforesaid formula is adjusted:

$$\Delta x_i = (H - G - (H + G + 10)x_i) / (H + G + 11)$$

$$x_i = x_i + \Delta x_i$$

This is an iterative formula of the state potency value. Only the instructive guidance, rather than the accurate potency value, is needed. Thus, the iteration number *k* is set to 20 and the potency value matrix of each state is obtained.

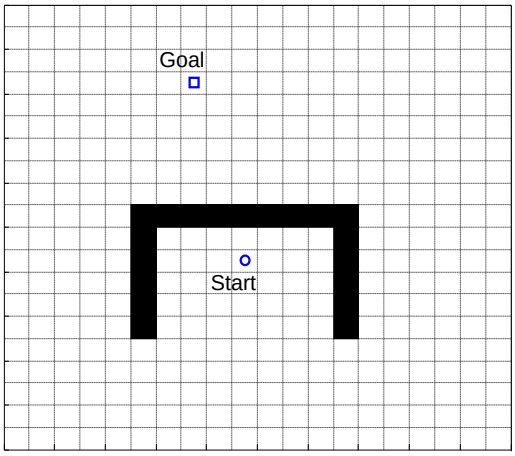



**Figure 3: Map Information of Concave Obstacles**

Figure3 is a grid map in the environment with concave obstacles. Wherein, the state which the diamond is located at is the target state and the state which the circle is located at is the starting state. Figure4 is the state potency value map after iterations of 20 times. The target state is located at the top of the mountain, i.e., the point with the largest potency value. The area where obstacles are located is a barranco, i.e., the point with the smallest potency value.

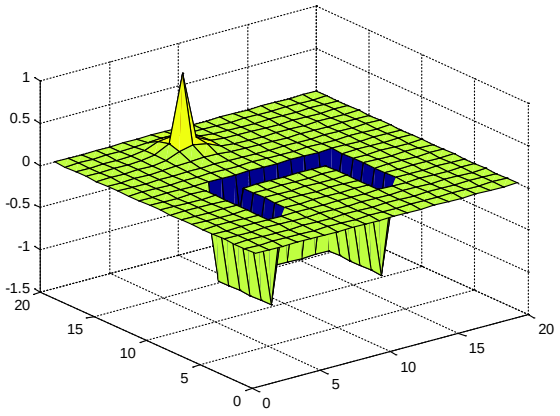



**Figure 4:** Figure of State Potency Value

The definition of reward value is as follows: $R(s,a)=100*(X(s')-X(s))$.

### *3.2. Improved Q-learning (INQL) Algorithm based on Neural Network*

After the basic principle of Q-learning algorithm is introduced, the previously proposed heuristic control strategy, new search strategy and MDP model of the path planning of the mobile robot can be applied in Q-learning algorithm, and then the Neural Network Based Improved Q-learning (INQL) Algorithm is proposed. Specific steps of the algorithm are as follows.

### *3.3. The Dynamic Path Planning Method Based on INQL*

In the INQL algorithm, the mobile robot can obtain the table of optimal Q values by making use of the environmental information, locations of the starting point and the target point. After the learning process ends, the mobile robot selects and executes the optimal action in each state by referring to the table of optimal Q values till it arrives at the target position. The track left by the mobile robot when it passes through the state is just the feasible path. After the feasible path is obtained, the cubic B-spline curve can be used to conduct the smooth fitting and a smooth path is then generated. When a moving obstacle is encountered in the environment, the mobile robot can consider the moving obstacle as the static obstacle. Then the obstacle-avoiding path planning is conducted.

In Table 2, Step 6 and Step 7 are treatment methods for new static obstacles. After the robot executes its selected optimal action, if the obstacle appears, Q value of the optimal action in the current state is very small negative integer and then the optimal action is reselected. Step 10, 11 and 12 are treatment methods of mobile obstacles, the robot shall observe and master the mode of motion of mobile obstacles within the visual range in real time, including the rate of motion and the direction of motion, and then take it as the static obstacle in accordance with the possible motion track. The specific process is shown in Figure 5.

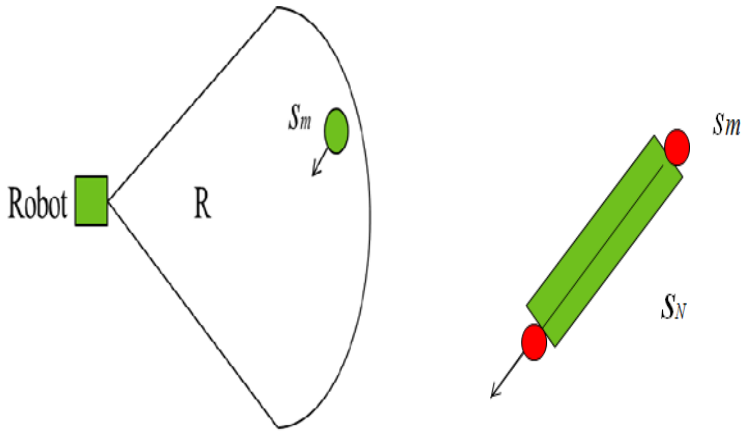



**Figure 5:** Illustration of Mobile Obstacle Treatment

**Table 1:** The INQL Algorithm

**The INQL Algorithm**

1: **Input:** Learning Rate $a$ ; Discount Factor $\gamma$ ; Convergence Parameter $\xi$ ;

Environmental Model G, Starting Point $s_i$ and Target Point $s_g$ ;

Maximum Number of Iterations $N$ ;

2: Initialization: Q value is 0 in all states; $s = s_i$ ;

3: While （$\|Q_t - Q_{t-1}\| > \xi$ && $t < N$ ）

4**:** repeat

4: If $Q(s,:)$ is 0, *a refers to* any *action for meeting the instructive control strategy;*

5: Otherwise any generated figure $0 < p < 1$ ;

6: If $p < \varepsilon$ , *a refers to any action* for *meeting the instructive* control strategy;

7: Otherwise *a* refers to the action selected with the searching strategy of Boltzmann;

8: Execute Action a to arrive at State $s'$;

9: If $s'$ refers to an unreasonable state, $Q(s,a) = Q(s,a) - P$ ;

10: If $s'$ is the target state, $Q(s,a) = Q(s,a)+P$ ;

11: Under other conditions,

$$Q_t(s,a) = \begin{cases} (1-a)Q_t(s,a) + a\left[R(s,a) + \gamma \max Q(s',a)\right]; & s = s_t, a = a_t \\ Q_{t-1}(s,a); \quad others \end{cases};$$

12: Update table of optimal Q values;

13: End

14: Be back to the table of optimal Q values;

Wherein, $P$ refers to the positive integer termed as a penalty coefficient; $Q(s,:)$ refers to the array collection of $Q$ values of all State-Action Pairs in State $s$.

R refers to the visual range of the mobile robot, $s_m$ refers to the mobile obstacle and $s_N$ refers to the static obstacle extending to the possible impact point along the motion direction of the mobile obstacle.

**Table 2:** Dynamic Path Planning Method Based on INQL

| **Dynamic Path** Planning **Method Based on INQL** |
|---|
| 1: Input: Environmental Information G; Starting Point $s_i$ and Target Point $s_g$; |
| 2: Initialization: $Q$ values in the table are all 0. |
| 3: Learning Stage: The table of optimal $Q$ values is obtained; |
| 4: **Initialization of Planning** Stage**:** The path sequence is initialized to 0; $s = s_i$; |
| 5: While ( $s \neq s_g$ ) ; |
| 6: repeat |
| 7: There are no mobile obstacles within the mobile robot's visual range: $R$ |
| 8:If $s \neq s_o$, the action with the largest Q value is selected and executed at State $s$ in accordance with the table of Q values till it arrives at the new Stage $s'$; if there are multiple actions with the largest Q value, the one with the smallest angle between it and the last action shall be selected; |
| 9: If $s'$ is the static obstacle absent from the learning process, $Q(s,a) = -PR$ , $s = s_0$; |
| 10: The new optimal action shall be reselected after being back to the last state: $s_0$ |
| 11: There is any mobile obstacle within the mobile robot's visual range $R$; |
| 12: The direction of motion and the rate of motion of mobile obstacles $s_m$ are observed to estimate the possible running track $l$ and impact point $C$; |
| 13: The mobile obstacle $s_m$ is pulled to a possible impact point $C$ along the possible motion track $l$ and finally becomes a static obstacle $s_N$; |
| 14: The mobile obstacle $s_m$ is taken as a new static obstacle $s_N$, the path is planned with Step (6) and Step (7) |

and meanwhile the cubic b spline curve is used for real-time fitting;

15: **End while**

16: Until $s = s_g$ ;

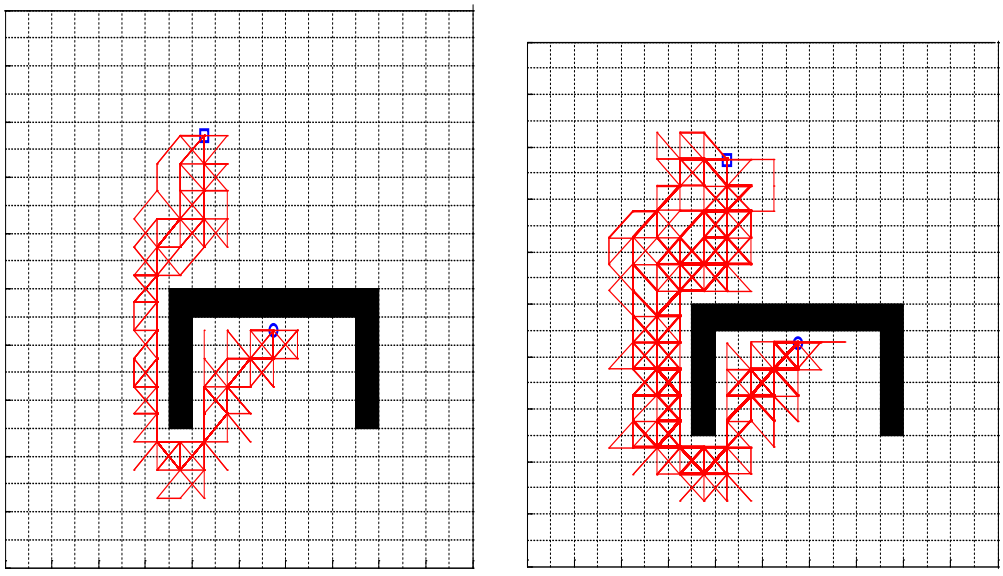
17: The cubic b spline curve is used for the real-time fitting for the obtained path;

18: **Output:** Smooth path $l_s$ ;

## 4. Simulation Results and Analysis

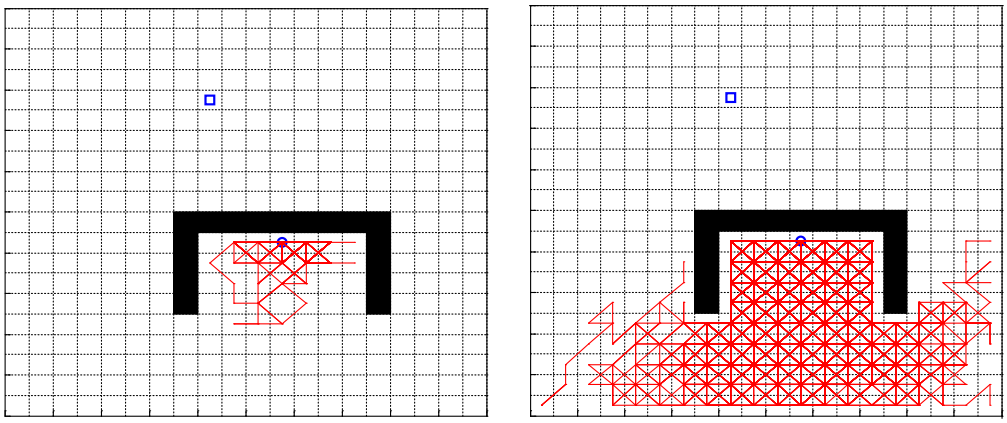
### *4.1. Performance Analysis of INQL Algorithm*

In this part, we will conduct the experimental analysis on the performance of INQL learning algorithm. The classic Q learning algorithm, the improved Q learning algorithm in [3] are comparatively analyzed. For the convenience of representation, the improved Q learning algorithm in [3] is recorded as IQL and the classic Q learning algorithm is recorded as CQL. The optimal Q table with respect to the environment illustrated in Figure3 is obtained. The three algorithms are used to analyze the difference of the learning rate. In the environment shown in Figure3, the obstacle is the concave obstacle and the initial location of the robot is at the bottom of the concave obstacle. Its target location is the back of the concave obstacle.



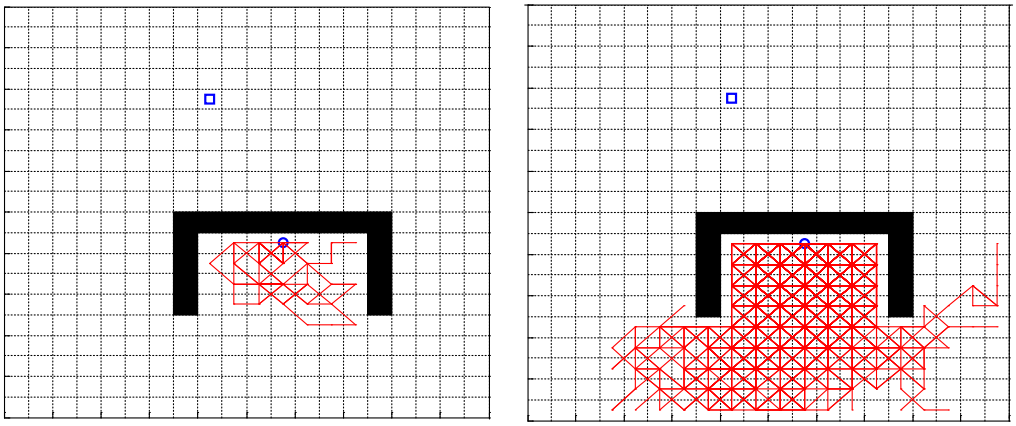
**(a) Process of 100 times of learnings** **(b) Process of 500 times of learnings**

**Figure 6:** Effect picture of the INQL algorithm after 100 times of learnings and 500 times of learnings



**(a) Process of 100 times of learnings** **(b) Process of 500 times of learnings**

**Figure 7:** Effect picture of the IQL algorithm after 100 times of learnings and 500 times of learnings



**(a) Process of 100 times of learnings** **(b) Process of 500 times of learnings**

**Figure 8:** Effect picture of the CQL algorithm after 100 times of learnings and 500 times of learnings

Figure6, Figure7 and Figure8 are tracking of states covered by the robot during the process of moving after 100 times and 500 times of learning with the INQL, CQL and IQL algorithms. In the figures above, circles refer to the start positions, diamonds refer to target positions and red lines refer to tracks in the state where the mobile robot passes through during the learning process.

In Figure6, after 100 times of learnings, the robot has arrived at the target point for several times after 100 times of learnings, the number of states that it passes through is small and tracks are characterized in centralized distribution. In Figure7 and Figure8, after 100 times of learnings, the robot doesn't go out of the region with concave obstacles. After 500 times of learnings, although it goes out of the region with obstacles, it has never arrived at the target position and the robot is far from the target point.

In order to verify the effectiveness of INQL algorithm, the stimulation test is conducted separately with different learning methods to solve the same problem in the same environment. Figure9 is the comparison chart of convergence rate of three algorithms. Wherein, the abscissa refers to the number of learnings and the ordinate refers to the number of steps that the robot makes. During each learning cycle, if the robot arrives at the target position, the learning process is end. If the robot does not arrive at the target position or bump into any obstacle after 100 steps, the learning process is also terminated. The maximum number of learning cycles is set to 4000. In order to make the figure seem more clear and reflect the trend of gradual decrease of the number of steps with the enlargement of the cycle, Figure9 is drawn based on the average value of steps taken in the adjacent fifty cycles.

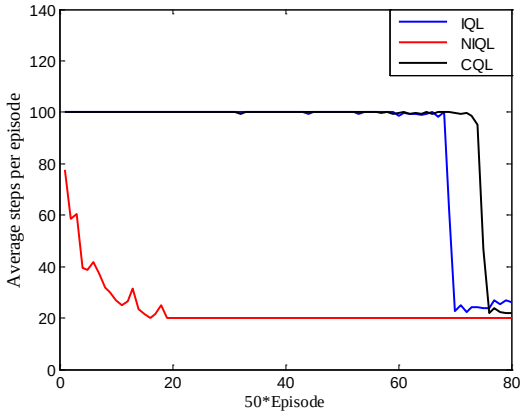



**Figure 9:** Comparison of Convergence Effects of Three Q Learnings

As shown in Figure9, the convergence of the INQL algorithm is fastest and is stably maintained after about 1000 times of learning cycles. The reason is that the potency value obtained with the NN model proposed in [15]. The instructive reward function based on the NN model is designed in the INQL algorithm. Thus, there is relatively good effect on such complex environmental information. IQL algorithm reaches a steady state after 3500 learning cycles. CQL algorithm has the worst performance and reaches a steady state after about 3800 learning cycles. The number of steps taken when each of the two algorithms are used to reach the steady state is larger than that when INQL algorithm is used.

In order to evaluate the performance of path planning method based on INQL, we can give the planned results in static and different dynamic environments, compared with the CQL and IQL algorithms.

### *4.2. Path Planning in Static Environment*

The path planning results based on the INQL, CQL, IQL learning algorithms in the environment with concave obstacles shown in Figure2 are illustrated in Figure10. Besides, an improved RRT method is also used for the path planning task.

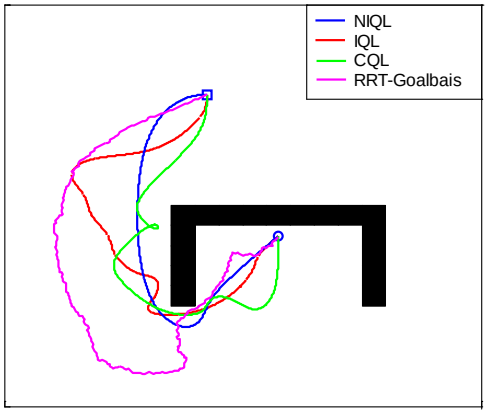



**Figure 10:** Planned Results of Environment with Concave Obstacles Based on Different Algorithms

Figure10 refers to paths planned based on RRT-Goalbais algorithm [16] and different Q learnings.. Paths shown in the figure are all smooth paths after the cubic b spline curve fitting. As illustrated in Figure10, the path planned based on INQL is relatively smooth, which is caused by the direction angle limitation for the reward function. Paths planned based on IQL and CQL are characterized in a large number of wide direction angles. All of aforesaid conditions are not applicable for the actual application of the mobile robot. The path planned based on RRT-Goalbais algorithm are characterized in the longest distance and worst smoothness. Secondly, the length of the path planned with the method introduced by the paper is optimal, with the longest safe distance with the obstacle.

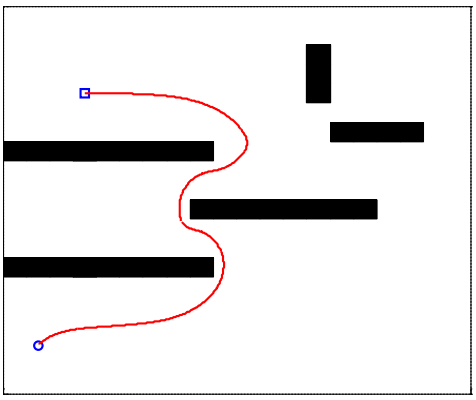
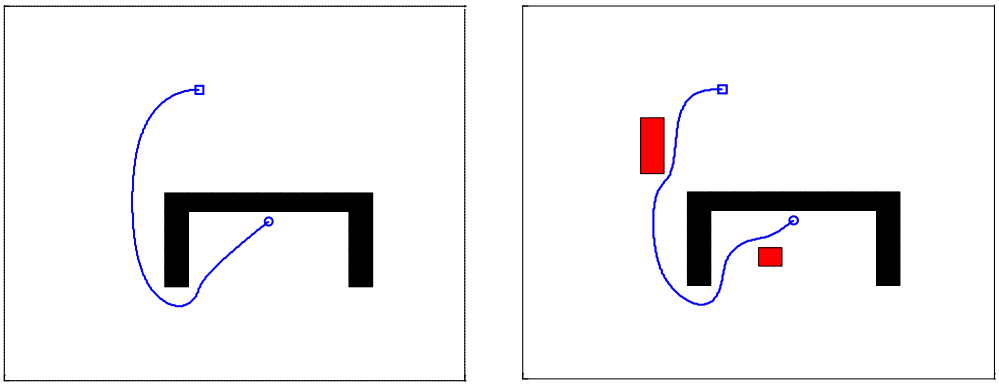


**Figure 11:** Planned Result in Narrow Passage

In order to verify the universality of the planning method, Figure11 refers to the planned result in the environment with the narrow passage based on INQL. As illustrated in Figure11, the planned path smoothly gets through two narrow passages, both keeping a safe distance from the obstacle and maintaining the optimal path distance.

### *4.3. Path Planning in Uncertain Environment*

In the actual application, the detailed information of the environment cannot be completely obtained in many times or the location of the obstacle cannot be completely determined. The robustness of the path planning algorithm is required.



**(a) Path in certain environment** **(b) Path in uncertain environment**

**Figure 12:** Planned Result in Uncertain Environment

In order to verify the effectiveness of the proposed dynamic planning method in the uncertain environment, the new obstacle is set up in the environment with concave obstacles in Figure2 on the previously planned path as shown in the red region of Figure12 (b). Wherein, the optimal Q value indicates the result obtained through the learning shown in Figure6 and there is no need for additional learning.

As shown in Figure12(b), the planning method based on INQL can keep away from new obstacles without relearning, due to that the planning method based on INQL learning is adopted on the basis of table of Q values. The table of optimal Q values stores Q values of all actions at each state. When a new state with an obstacle is reached a sub-optimal action will be reselected till a state without obstacles is reached. When other planning methods such as A*, RRT and PRM are adopted, the re-planning is required which is a disadvantage compared with the path planning method based on INQL.

### *4.4. Path Planning When Initial Position is Changed*

Another advantage of the proposed INQL method is that the method is still applicable when the initial position is changed. Other planning methods require re-planning from the new initial position. The experimental result is shown in Figure13. The mobile robot can still start from another initial position and arrive at the final position via a smooth path, with a safe distance from the obstacle and quasi-optimal length of the path. The effectiveness of the proposed dynamic planning method is verified when the initial position is changed.

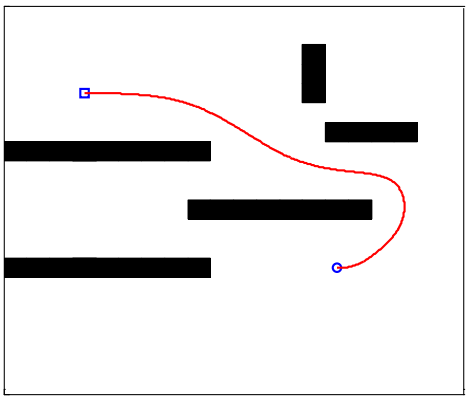


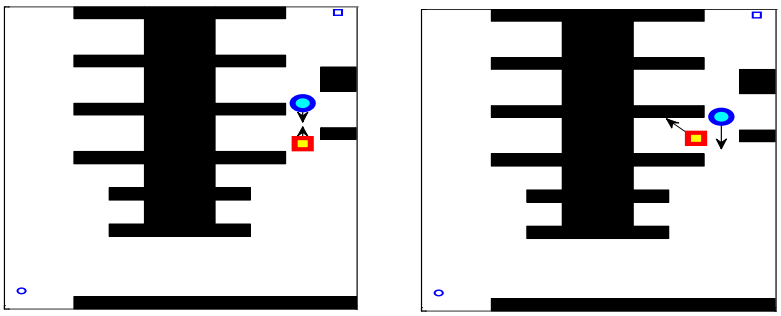
**Figure 13:** Planned Result of Random Initial Position

### *4.5. Path Planning in Environment with Dynamic Obstacles*

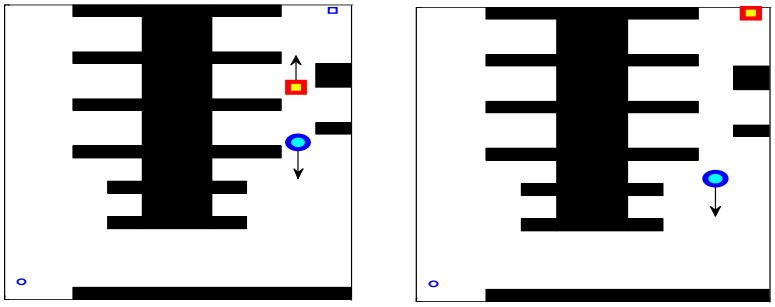
In order to verify the planning effect of the method in the environment with dynamic obstacles, experiments are conducted in our laboratory environment as shown in Figure14. The robot's initial position is the position of the door and the target position is down at the far end of the corridor by the window. Static obstacles, e.g., experimental desk, chair, beam, etc. are distributed in the environment.

**Figure 14:** The Laboratory Environment

The simulation result is as shown in Figure15. The blue circle and diamond represent the initial position and the final position, respectively; the red diamond represents the mobile robot and its motion speed is $0.5m/s$; the black region represents the static obstacle, the blue circle represents the moving obstacle with $0.3m/s$ speed.



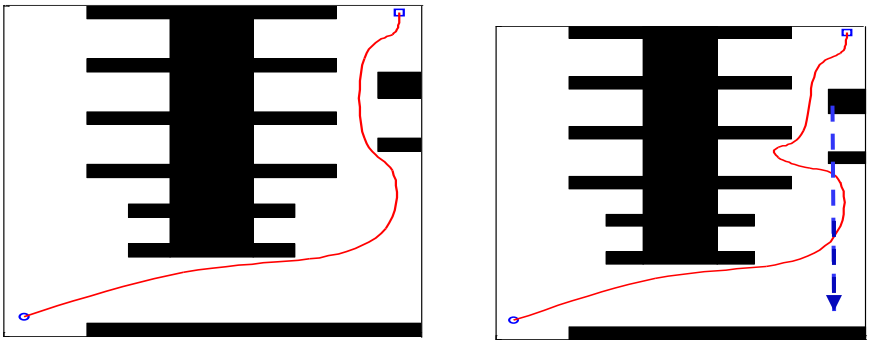
**(a)Locating obstacle** **(b) To avoid obstacle**



**(c) Successfully avoiding obstacle** **(d) Arriving at the end**

**Figure 15:** Simulation result of environment with dynamic obstacles

As illustrated in Figure15, the robot successfully avoids the obstacle and arrives at the target position in the laboratory environment, which verifies the effectiveness of the proposed dynamic planning method. Figure16 shows the robot's motion tracks in the environment without moving obstacles and with moving obstacles. In Figure16 (b), the red solid line represents the mobile robot's motion track. There is a process that the mobile robot avoids the obstacle.



**(a) In the environment without mobile obstacles** **(b) In the environment with mobile obstacles**

**Figure 16:** Robot's Motion Path

## 5. Conclusion

In the paper, the INQL algorithm is proposed by integrating the bio-inspired neural network model and the Q-learning algorithm. Compared with other Q-learning algorithms, the rate of convergence of the algorithm is significantly increased, especially in the complex environment. In addition, a dynamic path planning method based on INQL is proposed. Compared with traditional algorithms for path planning, the method can solve the difficultly solved problems with the change in the mobile robot's initial position and moving obstacles. In order to verify the effectiveness of INQL algorithm, different Q learning algorithms are comparatively analyzed in terms of learning rate in the concave environment. The simulation result indicates that the convergence rate of the INQL algorithm is significantly increased compared with other improved Q learning algorithms. In addition, in order to verify the effectiveness of the dynamic planning method, simulations and experiments are conducted in different dynamic environments. The results indicate the effectiveness of the proposed method.

## References


[1] Watkins C．Q-Learning[J]．Machine Learning，1992，8(3)：279-292.

[2] QIU Xue-na, LIU Shi-rong, SONG Jia-tao，Simon X. YANG. A Complete Coverage Path Planning Method for Mobile Robots in Uncertain Dynamic Environments[J].Robot. 2006,28(6):586-592.

[3] Amit Konar et al. A Deterministic Improved Q-Learning for Path Planning of a Mobile Robot. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 43, No. 5, 2013,1141-1153.

[4] I. Goswami, P. K. Das, A. Konar, and R. Janarthanan, "Extended Q-learning algorithm for path-planning of a mobile robot," in Proc. 8th Int. Conf. SEAL, Dec. 2010, pp. 379–383.

[5] Caihong Li, Ping Chen. Adaptive Behavior Design Based on FNN for the Mobile Robot. IEEE International Conference on Automation and Logistics. 2009. 1952-1956.

[6] Velappa Ganapathy, Soh Chin Yun. Neural Q-Learning Controller for Mobile Robot. IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Singapore, 2009.

[7] Zheng Qin and Jason Gu. Neural Q-learning in Motion Planning for Mobile Robot. Proceedings of the IEEE International Conference on Automation and Logistics, 2009.

[8] Zenhua Huang. Structured Enhancement Learning based on Manifold and its Application to Mobile Robot [D]. phd thesis, National University of Defence Technology, 2012.

[9] Zaobin Li. Reinforcement Learning in Autonomous Mobile Robot Navigation and Control [D]. phd thesis, National University of Defence Technology, 2010.

[10] Bain M, Sammut C. A Framework for Behavioural Cloning[C], Machine Intelligence 15. 1995: 103-129.

[11] Ng A Y, Russell S J. Algorithms for inverse reinforcement learning[C], Icml. 2000: 663-670.

[12] Ng A Y, Harada D, Russell S. Policy invariance under reward transformations: Theory and application to reward shaping[C], ICML. 1999, 99: 278-287.

[13] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Phys. Lond., vol. 117, pp. 500–544, 1952.

[14] S. Grossberg, "Contour enhancement, short term memory, and constancies in reverberating neural networks," Stud. Appl. Math., vol. 52, pp. 217–257, 1973.

[15] Simon X. Yang, Max Meng. Neural Network Approaches to Dynamic Collision-Free Trajectory Generation. IEEE Transactions on Systems, Vol. 31, No. 3, pp.302-318, 2001.

[16] S.m.LaYalle. Rapidly-exploring random trees: a new tool for path planning [R]. Technical Report TR98-11, Computer Science Dept, Iowa State University, 1998.

[17] L.P. Kaelbling, M.L. Littman, et al. Reinforcement learning: A survey[J]. Journal of Artificial Intelligence Research, 1996, 4:237-285.

[18] Qin, Zheng, J. Gu. Neural Q-learning in motion planning for mobile robot. IEEE International Conference on Automation and Logistics IEEE, 2009:1024 - 1028.

[19] Zong, Ziliang, et al. "Elevator group control algorithm based on residual gradient and Q-learning." SICE 2004 Annual Conference IEEE, 2004:329-331 vol. 1.

[20] Liu, Wei Peng, et al. "A dispatching algorithm for elevator group control system of recurrent neural networks based on q-learning." Journal of Hebei University of Technology , 2013.

[21] Gaskett C. Q-Learning for Robot Control[J]. phd thesis, australian national university, 2002.Kuzmin V.

[22] Connectionist Q-Learning in Robot Control Task[J]. Scientific Proceedings of Riga Technical University, 2002.

[23] Wang Y, Silva C W D. Assess Team Q-Learning Algorithm in a Purely Cooperative Multi-Robot Task[C], ASME 2007 International Mechanical Engineering Congress and Exposition. 2007:627-633.

[24] Wang Y, Silva C W D. A Modified Q-Learning Algorithm for Multi-Robot Decision Making[C], ASME 2007 International Mechanical Engineering Congress and Exposition. 2007:1275-1281.

[25] Krishnan V, Lakshmivarahan S. Probability and Random Processes[J]. Journal of the Royal Statistical Society, 2001, 40(1):164-165.